DETAILED ACTION

1. Claims 1, 4-22 are presented for the examination.

Continued Examination Under 37 CFR 1.114

A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 08/23/2010 has been entered.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

4. Claim 12 is directed to the method claims that would not qualify as a statutory process would be a claim that recited purely mental steps. Thus, to quality as a 101 statutory process, the claim should be positively recite the other statutory class (the thing or product) to which it is tied, for example by identifying the apparatus that accomplishes the method steps. Appropriate correction is required to add the computer performs the step of the methods.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1, 4, 5, 8-14, 17-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Voellm (US 7100172 B2) in view of Stern (US 7047537 B1) and further in view of Colletti(US 5990907 A).

As to claim 1, Voellm teaches a link(linking , col 3, ln 50-55), a function in a dynamic link library(the DLL functions , col 3, ln 50-55), a computer device(remote server 210, col 3, ln 42-50), a link between an application program and a function in a dynamic link library of a computing device(col 3, ln 50-55), a remapping component(the additional dispatch table 404 is a copy [remapping]of the original dispatch table (314, see FIG. 3) generated by the loader 310[component], col 5, ln 35-37/loader 310, col 4, ln 63-67 to col 5, ln 1-3), providing a remapping component arranged to provide(col 5, ln 35-37 , col 4, ln 63-67), a call by the application (call , col 5, ln 5-8), the function at an address location in a first dynamic link library(the addresses of each routine of each loaded DLL (e.g., 306, col 4, ln 65-67), in response to a call by the application program to link to the function at an address location in a first dynamic link library(col 4, ln 66-67/ col 5, ln 5-10), an address location for the function(The

dispatch table 314 includes entries (e.g., 316). Each entry includes the resolved addresses for each of the routines of the loaded DLLs. The resolved addresses in dispatch table 314 effectively operate as pointers, col 4, ln 55-65/Processing examines the pointer 308 of initial.dll to locate the dispatch table that lists the address of routine2. Pointer 308 directs processing to updated dispatch table 402, col 2, ln 44-57), a further dynamic linking library(dispatch table 314, col 4, ln 55-65/ the dispatch table 402, col 5, ln 44-57), an address location for the function in a further dynamic link library(col 4, ln 55-56/ col 5, ln 44-57), so as to enable the application programmed to link directly to the function in the further dynamic link library(processing examines the pointer 308 of initial.dll to locate routine2. Pointer 308 directs processing to updated dispatch table 402. Updated dispatch table 402 continues to effectively point to routine2 located in initial.dll 306. As a result, app.exe utilizes routine2 of initial.dll 306 to perform the function requested, col 6, ln 15-25), converting a call by the application program to link to the function at an address location in a first dynamic link library to an address location for the function in a further dynamic link library (The dispatch calls are effectively re-pointed to a new DLL other than the original DLL provided (e.g., 306) as illustrated in FIG. 4 below. The process for associating a new DLL with the application is further described in the discussion of FIG. 5 below, col 5, ln 9-15/ Updated dispatch table 402 includes new pointers to LWIO.dll 406 for routines corresponding to the LWIO functionality (e.g., routine1'). For example, app.exe 302 includes a reference to routine1. Processing examines the pointer 308 of initial.dll to find the dispatch table to locate routine1. Pointer 308 directs processing to updated dispatch table 402. However, at updated dispatch table 402, the address for routine1 has changed. The address for routine1 now points to routine1' located in LWIO.dll 406 rather than routine1 of initial.dll 306.

As a result, app.exe utilizes routine1 to perform the function requested rather than routine1 of initial.dll 306, col 5, ln 55-67/ the dispatch call is updated to point to the corresponding routine of the LWIO DLL. Accordingly, as the application is executed, col 7, ln 12-15).

Voellm does not explicitly teach using the relocation instruction to insert into the export data table the address location for the function in the further dynamic link library and a call by the application program to link to the function in the first dynamic link library jumps directly to the address location for the function in the further dynamic link library using the address location inserted into the export data table without the use of a sub-routine in the remapping component. Howerver, Stern teaches the relocation instruction to insert into the export data table the address location for the function in the further dynamic link library and a call by the application program to link to the function in the first dynamic link library jumps directly to the address location for the function in the further dynamic link library using the address location inserted into the export data table without the use of a sub-routine in the remapping component(a branch table is provided in each parent DLL to enable each child DLL to subsequently be called. For instance, the third DLL 112 shown in FIG. 1 has the option of calling the fourth DLL 114 and the sixth DLL 118 depending upon which chain is being executed (i.e., which interface initiated execution). FIG. 2A is a diagram illustrating the inclusion of a branch table 202 in a parent DLL, the third DLL 112 of FIG. 1. Each parent DLL may be created to include a dummy address associated with each possible child DLL that may be executed. More particularly, as shown, in a first entry 204 of the branch table 202, if the child DLL to be executed is determined to be the fourth DLL (DLL4), a branch or jump instruction is provided with a dummy address, shown here to be "XXX." Similarly, in a second entry 206 of the branch table 202, if the child DLL to be

executed is determined to be the sixth DLL (DLL6), a branch or jump instruction is provided with another dummy address, col 3, ln 65-67 to col 4, ln 1-15/ Upon loading of the DLLs, branch tables of the parent DLLs are updated as appropriate to replace dummy addresses with entry points of the appropriate child DLLs. FIG. 2B is a diagram illustrating the updating of the branch table provided in the parent DLL, the third DLL 112 of FIG. 1. As shown, the branch table 202 is updated to include an entry in the branch table 202 that identifies an entry point of one of the child DLLs. More particularly, the branch table 202 is updated such that the appropriate dummy address is replaced with an entry point of the corresponding child DLL. As shown in FIG. 2B in the first entry 204, when the child DLL to be called is the fourth DLL, DLL4, and the jump address is "1000." However, as shown in the second entry 206, when the child DLL to be called is the sixth DLL, DLL6, the jump address is "2000." Thus, when the parent DLL is shared by two or more executable chains of DLLs, each child DLL is associated with one of the executable chains. In other words, it is necessary to specify which entry in the branch table is to be executed next when more than one entry exists in the branch table. This may be accomplished, for example, by associating a particular parameter with each of the executable chains, col 4, ln 15-40/ Once the chain is built, the set of DLLs may be executed without requiring a main program (e.g., parent code module) responsible for calling each of the DLLs. Moreover, since the chain is built when the DLLs are loaded, the DLLs can be modified without requiring recompilation, col 5, ln 3-10).

It would have been obvious to one of the ordinary skill in the art at the time the invention was made to modify the teaching of Voellm with Stern to incorporate the feature of the relocation instruction to insert into the export data table the address location for the function in

the further dynamic link library because this allows the code modules need not be called by a main program, the time that is typically required to call each code module and return to the main program is eliminated.

Voellm and Stern do not explicitly teach modify the export data table, wherein the relocation instruction modifies the export data table at least in part by inserting into the export data table the address location for the function in the further library. However, Colletti teaches modify the export data table, wherein the relocation instruction modifies the export data table at least in part by inserting into the export data table the address location for the function in the further library(In addition, the font manager records[further library] the function addresses of the corresponding native DLL functions for future use, col 2, ln 55-61/ In a system in which the font manager is executing, as illustrated in FIG. 1, the address in the application's export table pointing to the CreateFont function is replaced with an address pointing to a replacement function provided in the font manager, col 3, ln 7-15/ the font manager begins execution and substitutes addresses for its own function in the export table, col 2, ln 53-55).

It would have been obvious to one of the ordinary skill in the art at the time the invention was made to modify the teaching of Voellm and Stern with Colletti to incorporate the feature of modify the export data table, wherein the relocation instruction modifies the export data table at least in part by inserting into the export data table the address location for the function in the further library because this processes the application's request with the enhanced capability of handling the virtually installed fonts.

As to claim 4, Voellm teaches the remapping component is arranged to provide the respective address locations of a plurality of functions in a plurality of further dynamic link

libraries (col 4, ln 63-67 to col 5, ln 1-5).

As to claim 5, Voellm teaches providing a plurality of the remapping components between the first dynamic link library and the further dynamic link library (col 4, ln 63-67 to col 5, ln 1-5).

As to claims 8-9, they are apparatus claims of claim 1; therefore, they are rejected for the same reason as claim 1 above.

As to claim 10, Voellm teaches one or more of the further dynamic link libraries comprise a remapping component (col 5, ln 39-45).

As to claim 11, It is an apparatus claim of claim 1; therefore, it is rejected for the same reason as claim 1 above. In additional, Voellm teaches common location reference is used to link to a plurality of functions (col 4, ln 66-67/ col 5, ln 5-10) and Colletti teaches a second function, second address location(function addresses of the corresponding native DLL functions, col 2, ln 55-61) for the same reason as claim 1 above.

As to claim 12, 13, 14, 17, 18, 19, they are an apparatus claims of claims 1, 4, 11; therefore, they are rejected for the same reasons as claims 1, 4, 11 above.

3. Claims 6-7, 15, 16 20, 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Voellm (US 7100172 B2) in view of Stern(US 7047537 B1) in view of Colletti(US 5990907 A) and further in view of Hammond (US 5974470 A).

As to claim 6, Voellm, Stern and Colletti do not teach the application program is arranged to link by ordinal number to the dynamic link library. However, Hammond teaches the

application program is arranged to link by ordinal number to the dynamic link library(As noted by Windows expert Brian Livingston in the Sep. 2, 9 and 16, 1996 issues of InfoWorld, Microsoft and other vendors provide a number of shared DLLs with their applications, and different versions of these DLLs may have the same name . When a user installs a new application that also uses one of these DLLs the application may install a version of a DLL that is older or different than the one the other applications support. For example, Mr. Livingston states that the DLL called "OLE2NLS.DLL" is used by Microsoft Office 4.0, 4.2, 4.2c, 4.3, 4.3c, Microsoft Excell 5.0, 5.0c, Microsoft Word 6.0, 6.0a, 6.0c, Microsoft PowerPoint 4.0, 4.0c, Microsoft Project 4.0, and Microsoft Visual FoxPro 3.0 is Version 2.01, col 3, ln 9-20).

It would have been obvious to one of the ordinary skill in the art at the time the invention was made to modify the teaching of Voellm, Stern, Colletti and Stern with Hammond to incorporate the feature of the application program is arranged to link by ordinal number to the dynamic link library because this provides multiple versions of the same-name DLL module, in order to enable several different applications to run at once.

As to claim 7, Hammond teaches the application program is arranged to link by name to the dynamic link library (col 3, ln 9-20) for the same reason as claim 6 above.

As to claims 15, 16, 20, 21, they are apparatus claims of claims 6, 7; therefore, they are rejected for the same reasons as claims 6, 7 above.

2. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over Voellm (US 7100172 B2) in view of Stern (US 7047537 B1) in view of Colletti(US 5990907 A) and further in view of Enns(US. 7627350) .

As to claim 22, Voellm , Stern and Colletti do not teach mobile communication device. However, Enns teaches mobile communication device (in a dynamic linked library ("DLL") that is written to function with a particular mobile computing device, col 2, ln 45-55).

It would have been obvious to one of the ordinary skill in the art at the time the invention was made to modify the teaching of Voellm, Stern and Colletti with Enns to incorporate the feature of mobile communication device mobile computing devices have the ability to view data from a variety of different sources simultaneously.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to LeChi Truong whose telephone number is (571) 272-3767. The examiner can normally be reached on 8 - 5.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Sough Hyung can be reached on (571) 272-6799. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR of Public PAIP. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIP system, contact the Electronic Business Center (EBC) at 866-217-9197(toll-free).

/LeChi Truong/

Primary Examiner, Art Unit 2194

LeChi Truong

November 8, 2010